小写英文字母一共27个。一个int型有32位，因此可以利用int中的一位来表示该字符串是否含有某一个字母，比如用最低位表示’a’,那么0X 0011 则表示字符串 “ab”|”ba”。然后判断两个字符串是否有相同字符则可以用计算出的二进制值进行与操作，如果没有相同的字母，说明他们的每一位都不同，与的结果为0，否则说明其存在至少一位相同，这一位上两个数都为1，因此导致与的结果不为0.

#### [318. 最大单词长度乘积](https://leetcode-cn.com/problems/maximum-product-of-word-lengths/)

给定一个字符串数组 words，找到 length(word[i]) \* length(word[j]) 的最大值，并且这两个单词不含有公共字母。你可以认为每个单词只包含小写字母。如果不存在这样的两个单词，返回 0。

示例 1:

输入: ["abcw","baz","foo","bar","xtfn","abcdef"]

输出: 16

解释: 这两个单词为 "abcw", "xtfn"。

示例 2:

输入: ["a","ab","abc","d","cd","bcd","abcd"]

输出: 4

解释: 这两个单词为 "ab", "cd"。

示例 3:

输入: ["a","aa","aaa","aaaa"]

输出: 0

解释: 不存在这样的两个单词。

来源：力扣（LeetCode）

链接：https://leetcode-cn.com/problems/maximum-product-of-word-lengths

著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。

Java代码如下：

class Solution {

public int maxProduct(String[] words) {

int len = words.length;

int[] Count = new int[len];

int ans = 0;

for(int i=0;i<len;i++)

for(int j=0;j<words[i].length();j++)

{

int temp = words[i].charAt(j) - 97;

Count[i] = Count[i]|(1<<temp);

}

for(int i=0;i<len;i++)

for(int j=i+1;j<len;j++)

{

if((Count[i]&Count[j])==0)

{

int t = words[i].length()\*words[j].length();

ans = ans>t?ans:t;

}

}

return ans;

}

}

Js 代码如下：

var maxProduct = function(words) {

let len = words.length,Count = Array(len).fill(0),ans = 0;

for(let i=0;i<len;i++)

{

for(let j=0;j<words[i].length;j++)

{

let n = words[i][j].charCodeAt() - 97;

Count[i] = Count[i]|(1<<n);

}

}

for(let i=0;i<len;i++)

for(let j=i+1;j<len;j++)

{

if((Count[i]&Count[j])===0)

{

let num = words[i].length\*words[j].length;

ans = ans>num?ans:num;

}

}

return ans;

};

### **8、交换两个数(不借助第三变量)**

不用第三个变量交换两个数的方法也有几种，例如a = a + b;  b = a - b; a = a - b。下面这种方法可以实现的基础是一个数m与另一个数n异或，再与n异或，得到的结果是m.

依据的是公式：

a^a = 0, 0^a = a,异或操作满足交换律和结合律

/\*

不适用临时变量，交换两个数

a = a ^ b

b = b ^ a

a = a ^ b

\*/

void mySwap(int\* a,int\* b){

(\*a) ^= (\*b) ^= (\*a) ^= (\*b);

}

### **9、求一个数的绝对值**

下面的方法实现的基础是将n右移31位，可以获得n的符号。

/\*

取绝对值n右移31位，可以获得n的符号。若n为正数，得到0；若n为负数，得到 -1，负数与-1异或恰好把符号位置0。为负数时，因为 a^-1 相当于abs(a)-1,所以要加上1，也就是-（-1）。只是-1的时候满足，其他负数情况未必满足。

\*/

int myAbs(int n){

return (n ^ n >> 31) - (n >> 31);

}

### **10、求两个数的平均值**

第一种方法较为普遍且简单，不多说了。第二种方法，需要知道的是，( m ^ n ) >> 1得到的结果是m和n其中一个数的有些位为1的值的一半，m & n得到的结果是m 和n都为1的那些位，两个结果相加得到m和n的平均数。

/\*

求m和n的平均数

\*/

int getAverage(int m,int n){

return (m + n) >> 1;

}

/\*

求m和n的平均数

(m ^ n) >> 1 -> 获得m和n两个数中一个数的某些位为1的一半

m & n -> 获得m和n两个数中都为1的某些位

\*/

int getAverage\_2(int m,int n){

return ((m ^ n) >> 1) + (m & n);

}